REMARKS

This response is submitted in response to an Office Action mailed on May 4, 2007. Claims 12-15, 21-25, and 30-34 are withdrawn from consideration. Claims 1-11, 16-20, 26-29, and 35-42 were pending at the time the Office Action was issued. Claim 4 is canceled. Claim 43 is hereby added. Applicants hereby amend claims 1, 3, 5-9, 11, 16-19, 26-29, and 35-41. Claims 1-3, 5-11, 16-20, 26-29, and 35-43 remain pending.

Applicants respectfully express their appreciation to Examiner Gelagay for her telephone message of July 10, 2007, in which the Examiner clarified that claims 26-29 are not withdrawn from consideration.

In the interest of reducing the issues to be considered in this response, the following remarks focus principally on the patentability of independent claims 1, 16, 26, 36, and 41. The patentability of each of the dependent claims is not necessarily separately addressed in detail. However, Applicants' decision not to discuss the differences between the cited art and each dependent claim should not be considered as an admission that Applicants concur with the conclusions set forth in the Office Action that these dependent claims are not patentable over the disclosure in the cited references. Similarly, Applicants' decision not to discuss differences between the prior art and every claim element, or every comment set forth in the Office Action, should not be considered as an admission that Applicants concur with the interpretation and assertions presented in the Office Action regarding those claims. Indeed, Applicants believe that all of the dependent claims patentably distinguish over the references cited. Moreover, a specific traverse of the rejection of each dependent claim is not required, since dependent

I. <u>Claim Objection</u>

Claim 29 is objected to because of informalities. Applicants have amended claim 29 to recite "includes an instruction." Accordingly, Applicants respectfully request reconsideration and withdrawal of this objection.

II. <u>Rejection Under 35 U.S.C. § 112</u>

Claims 3 is rejected under 35 U.S.C. §112, Second Paragraph, as being indefinite. Applicants have amended claim 3 and deleted "structure-related data." Accordingly, Applicants respectfully request reconsideration and withdrawal of this rejection.

III. <u>Rejections Under 35 U.S.C. § 101</u>

Claims 1-11 are rejected under 35 U.S.C. §101 because the claimed invention is directed to non-statutory subject matter. Claim 4 is canceled. Applicants have amended claim 1 to recite "executing the requested resource" and "preventing the requested resource from execution." Applicants respectfully submit that both of these limitations are concrete, tangible actions and constitute proper statutory subject matter. Accordingly, Applicants respectfully request reconsideration and withdrawal of the rejections to claims 1-3, and 5-11.

Additionally, Claims 35-40 are rejected under 35 U.S.C. §101 because the claimed invention is directed to non-statutory subject matter. Applicants have amended each of the claims 35-40 to recite a "computer-accessible storage medium," which again is proper statutory subject matter. Accordingly, Applicants respectfully request reconsideration and withdrawal of the rejections to claims 35-40.

IV. REJECTIONS UNDER 35 U.S.C. § 102

Claims 1-4, 10-11, 16, 20, 26, 35 and 38-42 are rejected under 35 U.S.C. § 102(b) as being unpatentable over U.S. Patent 6,188,885 to Garst et al. (hereinafter "Garst"). Respectfully, Applicants submit that the claims are allowable over Garst for at least the reasons explained in detail below.

Claims 1-4 and 10-11

Claims 2-4 and 10-11 depend from Claim 1. Claim 1 recites:

> 1. A method for managing access to resources, comprising:
>
> generating a list of resource signatures, each of the resource signatures being generated based at least on function names included in an import table of a corresponding resource;
>
> accessing the list of resource signatures, each of the resource signatures configured with an accessibility status, wherein the accessibility status includes one of loadable and restricted;
>
> generating a verification signature for a requested resource, the verification signature being generated based at least on function names included in an import table of the requested resource;
>
> comparing the verification signature for the requested resource to the list of resource signatures;
>
> executing the requested resource if the resource signature matches the verification signature and the accessibility status is loadable; and
>
> preventing the requested resource from execution if the resource signature matches the verification signature and the accessibility status is restricted.

Applicants respectfully assert that Garst does not teach or suggest every aspect of claim 1. First, Garst does not teach or suggest, "generating a list of resource signatures, *each of the resource signatures being generated based at least on function names included in an import table* of a corresponding resource," as recited in claim 1. (emphasis added).

Instead, Garst discloses generating a license key 610. (Column 5, Lines 3-6). The license key 610 is a digital signature that is "algorithmically derived from a license text string 600." (Column 5, Lines 22-24). Garst further discloses that the license text string 600 is a text string that "specifies the name of resource library vendor, the name of the program licensed to use the resource library, and the name of the resource library that has been licensed." (Column 5, Lines 15-20). In other words, Garst discloses deriving a digital signature from a text string. Accordingly, Garst does not teach or suggest "each of the resource signature being generated based at least on *function names included in an import table* of a corresponding resource," as recited in claim 1.

Second, since Garst does not teach or suggest, "each of the resource signature being generated based at least on *function names included in an import table* of a corresponding resource," Garst also cannot teach or suggest, "*accessing the list of resource signatures*, each of the resource signatures configured with an *accessibility status, wherein the accessibility status includes one of loadable and restricted*," as recited in claim 1. (emphasis added).

Third, Garst does not teach or suggest, "generating a verification signature for a requested resource, *the verification signature being generated based at least on function names included in an import table* of the requested resource," as recited in claim 1.

Instead, Garst discloses generating a license key 610. (Column 5, Lines 3-6). The license key 610 is a digital signature that is "algorithmically derived from a license text string 600." (Column 5, Lines 22-24). Garst further discloses that the license text string 600 is a text string that "specifies the name of resource library vendor, the name of the program licensed to use the resource library, and the name of the resource library that has been licensed." (Column 5, Lines 15-20). In other words, Garst discloses deriving a digital signature from a license text string 600. Accordingly, Garst does not teach or suggest, "*the verification signature being generated based at least on function names included in an import table* of the requested resource," as recited in claim 1.

Fourth, Garst does not teach or suggest, "*comparing the verification signature for the requested resource to the list of resource signatures*," as recited in claim 1. (emphasis added). Instead, Garst discloses using "license key 610 (digital signature) to verify the content of license text string 600," that is, verify that the text string 600 is from a resource library vendor. (Column 6, Lines 9-12). However, Garst's disclosure regarding using a license key 610 to verify a license *text string* does not teach or suggest comparing a digital signature with a list of other digital signatures. (emphasis added).

Fifth, since Garst does not teach the comparison of a digital signature with a list of other digital signatures, Garst cannot teach or suggest, "*executing* the requested resource *if the resource signature matches the verification* signature and the accessibility status is loadable," as recited in claim 1. (emphasis added). For the same reason, Garst also cannot teach or suggest, "*preventing* the requested resource from execution *if the resource signature matches the verification signature* and the accessibility status is restricted," as recited in claim 1.

Thus, for at least the above reasons, the method recited in claim 1 is not anticipated by Garst. Since claims 2-4 and 10-11 depend from claim 1, they are allowable over the cited reference to Garst at least due to their dependency, as well as due to additional limitations recited.

Claims 16 and 20

Claim 20 depends from claim 16. Claim 16, as amended, recites:

> 16. A method of restricting particular applications, comprising:
>
> receiving a list of application fingerprints corresponding respectively to restricted applications;
>
> receiving a request to execute an application;
>
> generating a confirmation fingerprint for the requested application, wherein the confirmation finger print is generated at least from function names included in an import table of the requested application;
>
> comparing the confirmation fingerprint to the list of application fingerprints; and
>
> restricting the requested application if the confirmation fingerprint matches one of the application fingerprints respectively corresponding to restricted applications.

Applicants respectfully assert that Garst does not teach or suggest every aspect of claim 16. First, Garst does not teach or suggest, "generating a confirmation fingerprint for the requested application, wherein *the confirmation finger print is generated at least from function names included in an import table of the requested application*," as recited in claim 16. (emphasis added).

Instead, Garst discloses generating a license key 610. (Column 5, Lines 3-5). The license key 610 is a digital signature that is "algorithmically derived from a license text string 600." (Column 5, Lines 22-24). Garst further discloses that the license text string 600 is a text string that "specifies the name of resource library vendor, the name of the program licensed to use the resource library, and the name of the resource library that has been licensed." (Column 5, Lines 15-20). In other words, Garst discloses deriving a digital signature from a license text string 600. Accordingly, Garst does not teach or suggest the above mentioned element of claim 16.

Second, since Garst does not teach or suggest, "wherein *the confirmation finger print is generated based at least on function names included in an import table of the requested application*," as recited in claim 16, Garst cannot teach or suggest, "*comparing the confirmation fingerprint to the list of application fingerprints*," as recited in claim 16. (emphasis added).

For the same reason, Garst also cannot teach or suggest, "restricting the requested application *if the confirmation fingerprint matches one of the application fingerprints* respectively corresponding to restricted applications," as further recited in claim 16. (emphasis added).

Thus, for at least the above reasons, the method recited in claim 16 is not anticipated by Garst. Since claim 20 depends from claim 16, it is allowable over the cited reference to Garst at least due to its dependency, as well as due to additional limitations recited.

Claim 26

Claim 26, as amended, recites:

> 26. An apparatus, comprising:
> an interface to receive a request for a running state of an application;
> an application identifier to generate an application digital signature for the application;
> an application manager to match the application digital signature against a list of stored digital signatures indicating whether corresponding applications are eligible or ineligible for a running state; and

an enabler to enable the running state for the application if the identifier is not matched to an identifier indicating that the application is ineligible.

Applicants respectfully assert that Garst does not teach or suggest every aspect of claim 26. Specifically, Garst does not teach or suggest, "*an application manager to match the application digital signature against a list of stored digital signatures* indicating whether corresponding applications are eligible or ineligible for a running state," as recited in claim 19. (emphasis added).

Instead, Garst discloses a "resource library licensing module 510" that "uses the license key to verify the content of license text string 600." (Column 6, Lines 9-12). Garst further discloses that license key is a digital signature, and that the license text string 600 is text that "specifies the name of the resource library vendor, and the name of the program licensed to use the resource library." (Column 5, Lines 15-20). In other words, Garst discloses the a resource library licensing module 510 that compares a digital signature with a text string, and not "the application digital signature against a list of stored digital signatures," as recited in claim 26. Thus, for at least the above reason, the apparatus recited in claim 26 is not anticipated by Garst.

Claims 35 and 38-40

Claims 38-40 depend from claim 35. Claim 35, as amended, recites:

35. A computer-accessible storage medium having an application programming interface (API), the API having one or more instructions to cause one or more processors to:
receive a request to run a program;
generate a digital signature for the program;
compare the generated digital signature against a compilation of digital signatures corresponding to restricted programs; and

> enable only those programs for which the signature does not match with any of the compiled signatures.

Applicants respectfully assert that Garst does not teach or suggest every aspect of claim 35. Specifically, Garst does not teach or suggest, "*compare the generated digital signature against a compilation of digital signatures corresponding to restricted programs*," as recited in claim 35. (emphasis added).

Instead, Garst discloses a "resource library licensing module 510" that "uses the license key to verify the content of license text string 600." (Column 6, Lines 9-12). Garst further discloses that license key is a digital signature, and that the license text string 600 is text that "specifies the name of the resource library vendor, and the name of the program licensed to use the resource library." (Column 5, Lines 15-20). In other words, Garst discloses the a resource library licensing module 510 that compares a digital signature with a *text string*, and not "the generated digital signature against a compilation of digital signatures," as recited in claim 35. (emphasis added).

Thus, for at least the above reason, the apparatus recited in claim 35 is not anticipated by Garst. Since claims 38-40 depend from claim 35, they are allowable over the cited reference to Garst at least due to their dependency, as well as due to additional limitations recited.

<u>Claims 41-42</u>

Claims 41-42, as amended, recites:

> 41. A license enforcement method, comprising:
> generating a digital signature for each of a plurality of applications;
> classifying each of the digital signatures in accordance with a licensing status for the corresponding applications;
> coding an operating system to:

> include the classified digital signatures,
> generate a digital signature for a requested application,
> compare the digital signature for the requested application to the classified digital signatures, and
> run the requested application when the digital signature for the requested application does not map to digital signature classified as not being licensed.

Applicants respectfully assert that Garst does not teach or suggest every aspect of claim 41. Applicants respectfully incorporate and reassert the argument present above in response to the rejection of claim 35 under 35 U.S.C. § 102(b) by analogy. Accordingly, Applicants assert that Garst does not teach or suggest, "compare *the digital signature for the requested application* to the *classified digital signatures*, as recited in claim 41. (emphasis added).

Moreover, since claim 42 depends from claim 41, it is allowable over the cited reference at least due to its dependency, as well as due to additional limitations recited.

V. REJECTIONS UNDER 35 U.S.C. § 103

Claims 5-9, 17-19, 27-28, and 36-37 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Garst in view of U.S. Patent 6,026,235 to Shaughnessy et al. (hereinafter "Shaughnessy"). Respectfully, Applicants submit that the claims are allowable over the cited references to Garst and Shaughnessy for at least the reasons explained in detail below.

Claims 5-9

Claims 5-9 depend from claim 1. Claim 1, as amended, recites:

1. A method for managing access to resources, comprising:

generating a list of resource signatures, each of the resource signatures being generated based at least on function names included in an import table of a corresponding resource;

accessing the list of resource signatures, each of the resource signatures configured with an accessibility status, wherein the accessibility status includes one of loadable and restricted;

generating a verification signature for a requested resource, the verification signature being generated based at least on function names included in an import table of the requested resource;

comparing the verification signature for the requested resource to the list of resource signatures;

executing the requested resource if the resource signature matches the verification signature and the accessibility status is loadable; and

preventing the requested resource from execution if the resource signature matches the verification signature and the accessibility status is restricted.

Applicants incorporate the argument presented above in response to the rejection of claim 1 under 35 U.S.C. §102(b) by analogy. Accordingly, Applicants first respectfully assert that Garst does not disclose, teach or fairly suggest, "generating a list of resource signatures, *each of the resource signatures being generated based at least on function names included in an import table of a corresponding resource*," as recited in claim 1. Instead, Garst discloses deriving a license key 610 (digital signature) from a license *text string* 600. (emphasis added). (Column 5, Lines 22-24).

Moreover, the deficiencies of Garst are not remedied by Shaughnessy. The disclosure in an assertedly anticipating reference must provide an enabling disclosure of the desired subject matter; mere naming or description of the subject matter is insufficient, if it cannot be produced without undue experimentation.

Elan Pharm., Inc. v. Mayo Foundation for Medical and Education Research, 346 F.3d 1051, 1054, 68 USPQ2d 1373, 1376 (Fed. Cir. 2003). MPEP § 2121.01.

In this instance, Shaughnessy discloses that the names and address locations of functions in an application may be located using import tables and "debug info." (Column 4, Lines 14-22). However, Shaughnessy does not teach or suggest "*generating a list of resource signatures*, each of the resource signature being generated based at least on *function names included in an import table* of a corresponding resource," as recited in claim 1. (emphasis added).

Moreover, obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so. *In re Kahn*, 441 F.3d 977, 986, 78 USPQ2d 1329, 1335 (Fed. Cir. 2006) (discussing rationale underlying the motivation-suggestion-teaching requirement as a guard against using hindsight in an obviousness analysis). In this instance, there is nothing in the teachings of either Garst or Shaughnessy, or any other teaching, suggestion, or motivation, to indicate that one would be inclined to generate a license key 610 using function names from an import table rather than a licensing text string 600.

For the same reasons, the cited references to Garst and Shaughnessy, whether individually or in combination, cannot teach or suggest, "generating a verification signature for a requested resource, *the verification signature being generated based at least on function names included in an import table of the requested resource*," as recited in claim 1. (emphasis added).

Second, as stated in the argument incorporated above, Garst discloses a "resource library licensing module 510" that "uses the license key to verify the content of license text string 600." (Column 6, Lines 9-12). Accordingly, Garst does not teach or suggest:

> *comparing the verification signature for the requested resource to the list of resource signatures*;
> executing the requested resource *if the resource signature matches the verification signature* and the accessibility status is loadable; and
> preventing the requested resource from execution *if the resource signature matches the verification signature* and the accessibility status is restricted. (emphasis added).

Moreover, the deficiencies of Garst are not remedied by Shaughnessy. Shaughnessy's disclosure is related to determining names and addresses of function from import tables. (Column 4, Lines 14-22). However, Shaughnessy does not disclose comparing a digital signature to other digital signatures.

Thus, for at least the above reasons, the cited references to Garst and Shaughnessy, whether individually or in combination, do not disclose, teach, or fairly suggest the method recited in claim 1. Since claims 2-5 depend from claim 1, they are allowable over the cited references at least due to their dependency, as well as due to additional limitations recited.

Specifically, claim 5 is further allowable over the cited references to Garst and Shaughnessy. Claim 5, as amended, recites:

> 5. A method according to Claim 1, wherein generating the verification signature for the requested resource includes:
> retrieving a plurality of names from the import table, wherein the plurality of names at least include function name;
> sorting the retrieved names;
> concatenating the sorted names; and
> executing a cryptographic manipulation of the concatenated names.

First, Garst does not teach or disclose, "*concatenating the sorted names*," as recited in claim 5. (emphasis added). Instead, Garst's disclosure is related to generating a license key 610 (digital signature) from a licensing text string 600. (Column 5, Lines 22-24). Moreover, the deficiencies of Garst are not remedied by Shaughnessy. Shaughnessy does not disclose the generation of digital signatures.

Second, Garst does not teach or disclose, "executing a cryptographic manipulation of *the concatenated names*," as recited in claim 5. (emphasis added). Instead, Garst discloses that generating a license key 610 (digital signature) from a *licensing text string 600* using a one way encryption process. (emphasis added). (Column 5, Lines 30-33). Moreover, the deficiencies of Garst are not remedied by Shaughnessy. Shaughnessy does not disclose the generation of digital signatures by an encryption process.

Thus, for at least the above reasons, the cited references to Garst and Shaughnessy, whether individually or in combination, do not disclose, teach, or fairly suggest the method recited in claim 5.

Claims 17-19

Claims 17-19 depend from claim 16. Claim 16, as amended, recites:

> 16. A method of restricting particular applications, comprising:
> receiving a list of application fingerprints corresponding respectively to restricted applications;
> receiving a request to execute an application;
> generating a confirmation fingerprint for the requested application, wherein the confirmation fingerprint is generated at least from function names included in an import table of the requested application;
> comparing the confirmation fingerprint to the list of application fingerprints; and

> restricting the requested application if the confirmation fingerprint matches one of the application fingerprints respectively corresponding to restricted applications.

Applicants incorporate the argument presented above in response to the rejection of claims 5-9 under 35 U.S.C. §103(a) by analogy. Accordingly, Applicants first respectfully assert that the cited references to Garst and Shaughnessy do not disclose, teach or fairly suggest, whether individually or in combination, "generating a confirmation fingerprint for the requested application, wherein *the confirmation fingerprint is generated at least from function names included in an import table* of the requested application," as recited in claim 16. (emphasis added).

Since claims 17-19 depend from claim 16, they are allowable over the cited references at least due to their dependency, as well as due to additional limitations recited.

Specifically, claim 17 is further allowable over the cited references to Grast and Shaughnessy. Claim 17, as amended, recites:

> 17. A method according to Claim 16, wherein generating a confirmation fingerprint for the requested application includes:
> Retrieving a plurality of names from the import table, wherein the plurality of names include function names;
> sorting the retrieved names;
> concatenating the sorted names in a predetermined manner; and
> hashing the organized names.

First, Garst does not teach or disclose, "*concatenating the sorted names* in a predetermined manner," as recited in claim 17. (emphasis added). Instead, Garst's disclosure is related to generating a license key 610 (digital signature) from a licensing text string 600. (Column 5, Lines 22-24). Moreover, the deficiencies of

Garst are not remedied by Shaughnessy. Shaughnessy does not disclose the generation of digital signatures.

Second, Garst does not teach or disclose, "hashing the *organized names*," as recited in claim 17. (emphasis added). Instead, Garst discloses that generating a license key 610 (digital signature) from a *licensing text string 600* using a one way encryption process. (emphasis added). (Column 5, Lines 30-33). Moreover, the deficiencies of Garst are not remedied by Shaughnessy. Shaughnessy does not disclose the generation of digital signatures by an encryption process.

Thus, for at least the above reasons, the cited references to Garst and Shaughnessy, whether individually or in combination, do not disclose, teach, or fairly suggest the method recited in claim 17.

Claims 27-28

Claims 27-28 depend from claim 26. Claim 26, as amended, recites:

> 26. An apparatus, comprising:
> an interface to receive a request for a running state of an application;
> an application identifier to generate an application digital signature for the application;
> an application manager to match the application digital signature against a list of stored digital signatures indicating whether corresponding applications are eligible or ineligible for a running state; and
> an enabler to enable the running state for the application if the application digital signature is not matched to a stored digital signature indicating that the application is ineligible.

Applicants incorporate the argument presented above in response to the rejection of claim 26 under 35 U.S.C. §102(b) by analogy. Accordingly, Applicants first respectfully assert that the cited reference to Garst do not teach or suggest, "an application manager to match the *application digital signature against a list of stored digital signatures* indicating whether corresponding applications are eligible or ineligible for a running state," as recited in claim 35. (emphasis added).

Moreover, the deficiencies of Garst are not remedied by Shaughnessy. Instead, Shaughnessy's disclosures are related to locating the names and address locations of functions in an application by using import tables and "debug info." (Column 4, Lines 14-22). However, Shaughnessy does not disclose the matching of digital signatures.

Additionally, since the cited references to Garst and Shaughnessy do not disclose, teach, or fairly suggest, "an application manager to match the *application digital signature against a list of stored digital signatures* indicating whether corresponding applications are eligible or ineligible for a running state", they also cannot disclose, teach or fairly suggest, "an enabler to enable the running state for the application *if the application digital signature is not matched to a stored digital signature indicating* that the application is ineligible," as recited in claim 26.

Accordingly, for at the above reasons, the cited references to Garst and Shaughnessy, whether individually or in combination, do not disclose, teach, or fairly suggest the method recited in claim 26. Since claims 27-28 depend from claim 26, they are allowable over the cited references at least due to their dependency, as well as due to additional limitations recited.

Specifically, claim 27 is further allowable over the cited references to Garst and Shaughnessy. Claim 27, as amended, recites:

> 27. An apparatus according to Claim 26, wherein the application identifier is to generate an application digital signature using an import table from the application.

Applicants respectfully assert that the cited reference to Garst do not teach or suggest an "application identifier is to *generate an application digital signature using an import table from the application*," as recited in claim 35. (emphasis added). In contrast, Garst discloses deriving a license key 610 (digital signature) from a license *text string* 600. (emphasis added). (Column 5, Lines 22-24).

Moreover, the deficiencies of Garst are not remedied by Shaughnessy. Instead, Shaughnessy's disclosures are related to locating the names and address locations of functions in an application by using import tables and "debug info." (Column 4, Lines 14-22). However, Shaughnessy does not teach the generation of digital signatures.

Thus, for at least the above reasons, the cited references to Garst and Shaughnessy, whether individually or in combination, do not disclose, teach, or fairly suggest the method recited in claim 27.

Moreover, claim 28 is further allowable over the cited references to Garst and Shaughnessy. Claim 28, as amended, recites:

> 28. An apparatus according to Claim 27, wherein the application identifier is to:
> retrieve an import table from an executable of the application;
> sort and string together the function names from the import table; and
> hash the stringed function names.

First, Garst does not teach or disclose, "sort and *string together* the function names from the import table," as recited in claim 28. (emphasis added). Instead, Garst's disclosure is related to generating a license key 610 (digital signature) from a licensing text string 600. (Column 5, Lines 22-24). Moreover, the deficiencies of Garst are not remedied by Shaughnessy. Shaughnessy discloses sorting "symbolic information" by name and address. (Column 4, Lines 25-27). However, Shaughnessy does not disclose *stringing together* function names. (emphasis added).

Second, Garst does not teach or disclose, "hashing the *stringed function names*," as recited in claim 28. (emphasis added). Instead, Garst discloses that generating a license key 610 (digital signature) from a *licensing text string 600* using a one way encryption process. (emphasis added). (Column 5, Lines 30-33). Moreover, the deficiencies of Garst are not remedied by Shaughnessy. Shaughnessy does not disclose hashing function names that have been stringed together.

Thus, for at least the above reasons, the cited references to Garst and Shaughnessy, whether individually or in combination, do not disclose, teach, or fairly suggest the method recited in claim 28.

Claims 36-37

Claims 36-37 depend from claim 35. Claim 35, as amended, recites:

> 35. A computer-accessible storage medium having an application programming interface (API), the API having one or more instructions to cause one or more processors to:
> receive a request to run a program;
> generate a digital signature for the program;
> compare the generated digital signature against a compilation of digital signatures corresponding to restricted programs; and
> enable only those programs for which the signature does not match with any of the compiled signatures.

Applicants incorporate the argument presented above in response to the rejection of claim 35 under 35 U.S.C. §102(b) by analogy. Accordingly, Applicants first respectfully assert that the cited reference to Garst do not teach or suggest, "*compare the generated digital signature against a compilation of digital signatures* corresponding to restricted programs," as recited in claim 35. (emphasis added).

Moreover, the deficiencies of Garst are not remedied by Shaughnessy. Instead, Shaughnessy's disclosures are related to locating the names and address locations of functions in an application by using import tables and "debug info." (Column 4, Lines 14-22). However, Shaughnessy does not teach the comparison of digital signatures.

Accordingly, for at the above reasons, the cited references to Garst and Shaughnessy, whether individually or in combination, do not disclose, teach, or fairly suggest the method recited in claim 35. Since claims 36-37 depend from claim 35, they are allowable over the cited references at least due to their dependency, as well as due to additional limitations recited.

Specifically, claim 37 is further allowable over the cited references to Garst and Shaughnessy. Claim 37, as amended, recites:

> 37. A computer-accessible storage medium according to Claim 36, wherein the one or more instructions to generate a digital signature for the program cause the one or more processors to hash a sorted list of function names from the import table.

Applicants incorporate the argument presented above in response to the rejection of claim 6 under 35 U.S.C. §103(a) by analogy. Specifically, Applicants respectfully assert that Garst discloses that generating a license key 610 (digital signature) from a *licensing text string 600* using a one way encryption process. (emphasis added). (Column 5, Lines 30-33). However, Garst does not teach or suggest, "hash a sorted list of *function names from the import table*," as recited in claim 37. (emphasis added). Moreover, the deficiencies of Garst are not remedied by Shaughnessy. Shaughnessy does not disclose the generation of digital signatures by an encryption process.

Accordingly, Applicants first respectfully assert that the cited references to Garst and Shaughnessy do not disclose, teach or fairly suggest the computer-accessible storage medium of claim 37.

<u>Claim 29</u>

Claim 29 is rejected under 35 U.S.C. § 103(a) as being unpatentable over Garst in view of Shaughnessy, and in further view of U.S. Patent 5,892,904 to Atkinson et al. (hereinafter "Atkinson").

Claim 29 depends from claim 26. Claim 26, as amended, recites:

> 26. An apparatus, comprising:
> an interface to receive a request for a running state of an application;
> an application identifier to generate an application digital signature for the application;
> an application manager to match the application digital signature against a list of stored digital signatures indicating whether corresponding applications are eligible or ineligible for a running state; and
> an enabler to enable the running state for the application if the identifier is not matched to an identifier indicating that the application is ineligible.

Applicants incorporate the argument presented above in response to the rejection of claims 27-28 under 35 U.S.C. §102(b) by analogy. Accordingly, Applicants respectfully assert that the cited references to Garst and Shaughnessy, do not disclose, teach or fairly suggest, whether individually or in combination, as recited in claim 26:

> an application manager to *match the application digital signature against a list of stored digital signatures* indicating whether corresponding applications are eligible or ineligible for a running state; and
>
> an enabler to enable the running state for the application *if the application digital signature is not matched to a stored digital signature* indicating that the application is ineligible. (emphasis added).

Moreover, the deficiencies of Garst are not remedied by Atkinson. Instead, Atkinson discloses using a hash function to sign .class files. (Column 20, Lines 1-3).

Accordingly, for at the above reasons, the cited references (Garst, Shaughnessy, and Atkinson) whether individually or in combination, do not disclose, teach, or fairly suggest the apparatus recited in claim 26. Since claim 28 depends from claim 26, it is allowable over the cited references at least due to their dependency, as well as due to additional limitations recited.

Specifically, claim 29 is further allowable over the cited references to Garst and Shaughnessy. Claim 29, as amended, recites:

> 29. An apparatus according to Claim 28, wherein the instruction to hash further includes an instruction to execute an MD5 hashing algorithm on the stringed function names.

As noted by the Examiner, the cited references to Garst and Shaughnessy, whether individually or in combination, does not teach or suggest executing an MD5 hashing algorithm. (Office Action, Page 10, Lines 1-3).

Moreover, the deficiencies of Garst are not remedied by Atkinson. Atkinson teaches using a MD5 hashing algorithm on ".class files," which is not equivalent to hashing "stringed function names." (Column 20, Lines 1-3).

Accordingly, Applicants first respectfully assert that the cited references to Garst and Shaughnessy do not disclose, teach or fairly suggest, "an instruction to execute an MD5 hashing algorithm *on the stringed function names*," as recited in claim 29.

VI. NEW CLAIM

Claim 43 is newly added. Claim 43 depends from and applies additional limitations to claim 1. Accordingly, claim 43 is allowable at least due to its dependency, as well for additional limitations recited in the claim.

CONCLUSION

Applicants respectfully submit that claims 1-3, 5-11, 16-20, 26-29, and 35-43 are in condition for allowance. If any issue remains unresolved that would prevent allowance of this case, the Examiner is requested to contact the undersigned attorney to resolve the issue.

Respectfully Submitted,

Dated: 8-3-07

By: [illegible]

Elliott Y. Chen
Reg. No. 58,293
Lee & Hayes, PLLC
421 W. Riverside Ave, Suite 500
Spokane, WA 99201
Phone: (206) 315-4001 x104
or (206) 315-7914
Fax: (206) 315-4004